

Martin S. Feather · Steven L. Cornford

Quantitative risk-based requirements reasoning

Received: 22 May 2002 / Accepted: 15 September 2002 / Published online: 25 February 2003
© Springer-Verlag London Limited 2003

Abstract At NASA we have been developing and applying a risk management framework, “Defect Detection and Prevention” (DDP). It is based on a simple quantitative model of risk and is supported by custom software. We have used it to aid in study and planning for systems that employ advanced technologies. The framework has proven successful at identifying problematic requirements (those which will be the most difficult to attain), at optimizing the allocation of resources so as to maximize requirements attainment, at identifying areas where research investments should be made, and at supporting tradeoff analyses among major alternatives. We describe the DDP model, the information that populates a model, how DDP is used, and its tool support. DDP has been designed to aid decision making early in development. Detailed information is lacking at this early stage. Accordingly, DDP exhibits a number of strategic compromises between fidelity and tractability. The net result is an approach that appears both feasible and useful during early requirements decision making.

Keywords Cost–benefit analysis · Decision making · Information visualization · Requirements · Risk · Tradeoffs

1 Introduction

At NASA we have been developing and applying our risk management framework, “Defect Detection and Prevention” (DDP), for several years. DDP has been designed to aid decision making during the earlier phase of advanced technology and system development. This is an important but challenging time of the life cycle. It is

important because these early decisions have the most leverage to influence the development to follow. It is challenging because information on which to base those decisions is incomplete and uncertain, and in the case of advanced technologies and systems there is little past experience from which to extrapolate.

As a design matures there are other decision support techniques that better capitalize upon knowledge of design details. For example, probabilistic risk assessment techniques (e.g., fault tree analysis, Bayesian methods) compute overall system reliability from design knowledge of how the system is composed of those components and estimates of individual component reliabilities.

Where there has been past experience at developing similar systems, parametric models (e.g., COCOMO for cost and schedule estimation) can be useful for making predictions for the project at hand. Eventually this expertise can be cast into “product families”.

In contrast, DDP aims to fill the niche of early decision making for advanced technology and system development. It does so by using a simple quantitative model of risk as the foundation for reasoning. The risk model itself, the information that populates it, the processes we follow to apply it, and the tool support we give it are all closely coupled. They each exhibit strategic compromises from what might be thought of as a “perfect” solution (perfect model, perfect process, perfect tool). The focus of this paper is on the nature of and justification for these strategic compromises, and how they facilitate feasible useful early requirements decision making.

The paper is organized as follows:

- Section 2 describes the quantitative risk model on which DDP is founded.
- Section 3 describes the way DDP is used, including its lightweight process and modest tool support.
- Section 4 describes the representation of information that comprises a DDP model.
- Section 5 describes the visualizations that connect users to the DDP tool.

M.S. Feather (✉) · S.L. Cornford
Jet Propulsion Laboratory,
California Institute of Technology,
4800 Oak Grove Drive, Pasadena, CA 91109, USA
E-mail: Martin.S.Feather@Jpl.Nasa.Gov

- Section 6 presents related work and conclusions.
- The Appendix provides further details of the DDP model together with fragments of a detailed example.

2 DDP risk model

The simple quantitative model at the heart of DDP involves just three key concepts: “*Requirements*” (what it is that the system or technology is to achieve), “*Failure Modes*” (what could occur to impede the attainment of the Requirements), and “*PACTs*” (what could be done to reduce the likelihood and/or impact of Failure Modes; PACTs is an acronym of Preventions, Analyses, process controls, and Tests). Requirements are related to Failure Modes, and Failure Modes are in turn related to PACTs. Specifically, Requirements are quantitatively related to Failure Modes to indicate how much each Failure Mode, should it occur, *impacts* each Requirement. Failure Modes are quantitatively related to PACTs, to indicate how much of a Failure Mode-reducing *effect* the PACT, should it be applied, has on the Failure Mode.

In a DDP model, a set of PACTs achieves benefits (Requirements are attained because the Failure Modes that impact them are reduced by the selected PACTs) but incurs costs (the sum total cost of performing those PACTs).

More details are presented in the subsections that follow, and further information is to be found in the Appendix.

2.1 DDP concepts

The subsections that follow give the details of DDP’s key concepts: Requirements, Failure Modes and PACTs, and the Impact and Effect relationships between them.

2.1.1 Requirements

Requirements are whatever the system under scrutiny is to achieve, and the constraints under which it must operate. They can be “product” requirements on the system (e.g., functional behavior, run-time resource needs), and/or “process” requirements (on the development process itself, e.g., cost and schedule). Each requirement is assigned a *weight*, representing its relative importance to mission success, etc.

2.1.2 Failure Modes

Failure Modes are all the things that, should they occur, will lead to loss of Requirements. Each Failure Mode is assigned an *a priori likelihood* (the chance of the failure mode occurring, if nothing is done to inhibit

it). Each Failure Mode is also assigned a *repair cost*: what it would cost to remove an instance of that Failure Mode from the system. If the DDP model makes a distinction between different time phases, the repair cost may be different for each of those possible time phases.

2.1.3 PACTs

PACTs are all the activities that could be done to reduce the likelihood of Failure Modes and/or reduce their impact on Requirements. These include *preventative measures* (e.g., training, standards, selection of high-quality parts), *detections* that discover instances of Failure Modes through analysis or test (e.g., code walkthroughs) so that those detected failure modes can be corrected prior to release/use, and *alleviations* that reduce the severity of failure modes (e.g., defensive programming that checks its inputs to ensure they are within specified bounds). We henceforth refer to these different kinds of PACTs as “prevention”, “detection”, and “alleviation” PACTs.

Each PACT is assigned a *cost*: the cost of performing it. Cost may be a measure of budget, schedule, physical attributes (e.g., weight and electrical power are predominant concerns for spacecraft), scarce resources (e.g., skilled personnel, high-fidelity testbeds), or indeed a mixture of these measurements. Each PACT is also assigned the time period within the development effort at which it would be performed (e.g., requirements phase, design phase).

It is possible that a PACT can *induce* a Failure Mode. For example, inserting error detection code can change the run-time behavior of a system, and so risk causing timing errors.

2.1.4 Impacts

For each Requirement x Failure mode pair, the “impact” is the proportion of the Requirement that would be lost if the Failure Mode were to occur. It is expressed as a number in the range 0 1; thus 0 means no impact whatsoever, and 1 means total loss of the Requirements. Note that a Failure Mode may impact multiple Requirements and do so to differing extents. Likewise, multiple Failure Modes may impact a Requirement, again to differing extents.

Impacts combine *additively*; e.g., if two different Failure Modes impact the same Requirement, then their combined impact on that Requirement is calculated as the sum of their individual impacts.

One seemingly strange consequence of our combination rule for impacts is that a Requirement can be *more* than completely impacted! For example, impacts of 0.8 and 0.7 on the same Requirement add up to a total impact of 1.5. This is in fact a useful measure of the amount of risk to be overcome in order to attain the Requirement. However, when assessing how much of

the Requirements have actually been attained, Requirements that are more than completely impacted contribute zero (not a negative amount, note).

2.1.5 Effects

For each PACT x Failure Mode pair, the *Effect* is the proportion by which the Failure Mode would be reduced if that PACT were applied. It is expressed as a number in the range 0–1; thus 0 means no reduction whatsoever, and 1 means total elimination of the Failure Mode.

Effects combine “*multiplicatively*”: when several PACTs reduce the same Failure Mode, their combined effect is computed as: $(1 - \text{the product, for each PACT } M_i, \text{ of } (1 - M_i \text{'s effect}))$.

Intuitively, PACTs act as “filters” arranged in series, such that each PACT filters out its effect’s proportion of the Failure Modes that enter it. See Fig. 1 for an example in which a PACT with an effect of 0.8 on some Failure Mode and another PACT with an effect of 0.3 on that same Failure Mode are each applied.

Note that the order in which these PACTs are applied does not matter.

As was the case for impacts, a PACT may effect multiple Failure Modes and do so to different extents, and a Failure Mode may be “effected” by multiple PACTs, again to different extents.

PACTs that *induce* Failure Modes are taken into account by having them *increase* the likelihood of those Failure Modes. Again, the degree of this influence is expressed as a number in the range 0–1. For a Failure Mode with likelihood L , and PACT with inducing effect of E , the new likelihood is calculated as $(1 - (1 - L) * (1 - E))$. Intuitively, if the Failure Mode was going to occur anyway or it is induced by the PACT (or both), then it will occur. Since the likelihood of $(P1 \text{ or } P2) = (1 - (\text{Likelihood of } P1) * (1 - \text{Likelihood of } P2))$, we get the formula above. Thus at the extremes for the PACT’s inducing effect E , 0 means no increase, and 1 means increase to certainty.

For example, if $L = 0.4$ and $E = 0.7$, this calculation is: $(1 - (1 - 0.4) * (1 - 0.7)) = (1 - 0.6 * 0.3) = 0.82$.

Note that it *does* matter in which order Failure Mode *reducing* PACTs interleave with Failure Modes *inducing* PACTs. For example, consider a “perfect” PACT (one that reduces a Failure Mode’s likelihood) and a Failure Mode inducing PACT. If the perfect PACT follows the inducing one, the Failure Mode will be eliminated conversely, the inducing PACT will cause the Failure Mode to occur after the point at which the perfect PACT has had a chance to apply. In practice, we assign PACTs to distinct time phases, and organize the calculations so that for PACTs of a given phase all the likelihood-reducing effects are calculated first (the relative order of which does not matter), and all the likelihood-increasing effects are calculated second (again, the relative order of which does not matter). This means that the Failure Modes induced within a time phase can be reduced only by PACTs of later time phases.

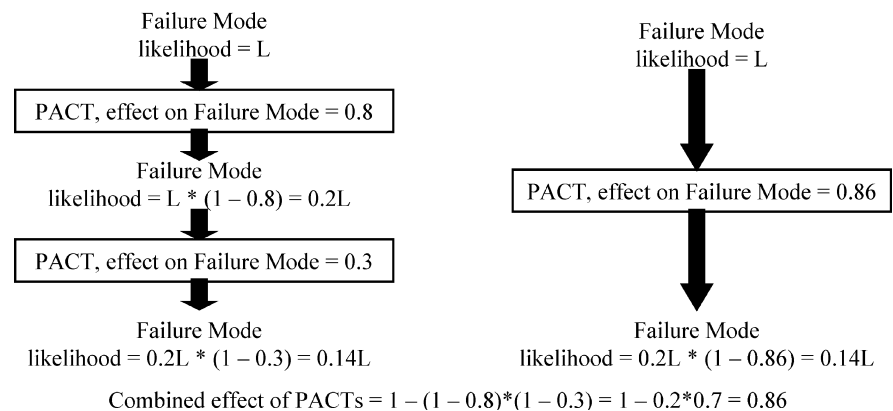
One of the reviewers made the suggestion that, when possible, the ordering of PACTs could be deliberately chosen to optimize their net effect. For example, given two PACTs that could be applied in either order, choose the ordering that puts first the PACT that induces Failure Modes. This would be an interesting extension to the current DDP implementation.

2.2 Calculations

In a DDP model, a set of PACTs achieves *benefits* (Requirements are met because the Failure Modes that impact them are reduced by the selected PACTs), but incurs *costs* (the sum total cost of performing those PACTs).

The measure of benefit of a DDP model is calculated as the sum of the weighted requirements’ attainment. The measure of cost of a DDP model is calculated as the sum of the costs of the PACTs selected for application, plus the sum of the costs of repairs of the Failure Modes that detection PACTs discover. Both of these measures take into account the detrimental impact of Failure Modes on Requirements, moderated by the effect of PACTs at reducing Failure Modes’ likelihoods and/or severities.

Fig. 1 DDP PACTs act like “filters” in series



The essential aspects are the calculation of Failure Modes' likelihoods and severities (in the course of which costs of PACTs and repairs are accumulated), followed by the calculation of Requirements' attainment. These are described next.

2.2.1 Failure mode likelihoods and severities

The calculation of each Failure Mode's likelihood starts from its a priori likelihood value. At each time phase, the effects on it of that phase's prevention and reduction PACTs reduce its likelihood. As discussed earlier, a PACT acts as a "filter" to remove some proportion of the Failure Mode. In the course of this calculation, the reduction in likelihood attributable to detection PACTs incurs a repair cost. This is the repair cost attributed to the Failure Mode at that phase, multiplied by the proportion by which the PACT reduces the Failure Mode's likelihood.

Example Consider a Failure Mode (e.g., a requirements flaw) that costs \$100 to repair at requirements formulation time. Suppose a PACT (e.g., requirements inspection) has an effectiveness of 0.7 against that Failure Mode. If the Failure Mode's likelihood prior to application of the PACT is 0.9, then after it will be $0.9 \cdot (1 - 0.7) = 0.27$. The reduction in likelihood is $0.9 - 0.27 = 0.63$, and so the repair cost is $\$100 \cdot 0.63 = \63 . An equivalent and more direct calculation of this is to simply multiply the Failure Mode's unit repair cost (\$100) by its likelihood prior to PACT (0.9) by the PACT's effect on that Failure Mode (0.7): $\$100 \cdot 0.9 \cdot 0.7 = \63 .

The PACTs of a time phase that *induce* failure modes are taken into account after all the PACTs of that phase that reduce failure modes. Their contribution is calculated using the combination rule discussed in the Effects subsection earlier.

The severity reductions attributable to alleviation PACTs are also calculated phase by phase, using the same kind of calculation as prevention PACTs, but decreasing Failure Mode severities rather than likelihoods.

2.2.2 Requirements attainment

The ideal requirements attainment is simply the sum of the weights of all the requirements. This ideal would only be achieved if all the failure modes were completely mitigated, by reducing their likelihoods and/or severities to zero.

The actual attainment of a requirement, taking into account Failure Modes and PACTs, is its weight $\cdot (1 - \text{the proportion to which it is at risk, capped at } 1)$. The proportion to which it is at risk is the sum over all Failure Modes of each Failure Mode's (likelihood \cdot severity \cdot impact on that Requirement). As mentioned earlier, this sum can exceed 1, hence the need to cap it at 1 in this calculation. The Failure Modes' likelihoods and

severities are calculated as described in the previous subsection.

Example Consider a requirement with weight 100 that is at risk due to two Failure Modes. Suppose that after taking PACTs into account the first Failure Mode has likelihood 0.9, severity 0.5, and impact 0.5, and the second has likelihood 0.4, severity 1.0, and impact 0.3. This requirement's attainment is thus $(100 \cdot (1 - ((0.9 \cdot 0.5 \cdot 0.5) + (0.4 \cdot 1.0 \cdot 0.3)))) = (100 \cdot (1 - (0.225 + 0.12))) = (100 \cdot (1 - 0.345)) = 65.5$.

3 DDP usage

3.1 Expert involvement

The success of a DDP application is crucially dependent on the involvement of experts. Their combined expertise must encompass:

Requirements:

- driving needs/goals/objectives (e.g., in our setting, the science mission objectives driving the need for an instrument's capabilities);
- environmental constraints on resources available to the system (e.g., RAM, power);
- environmental constraints on the extent to which the system can impact its environment (e.g., electromagnetic fields).

Failure Modes:

- development problems (inability to construct, test, repair, operate and maintain the system);
- the multitude of ways the operating system can fail to meet requirements.

PACTs:

- preventative measures that can be employed to reduce the likelihood of problems arising in the first place (e.g., coding standards, training, use of qualified parts);
- detections that can be employed to uncover the presence of problems prior to fielding and use of the system (e.g., inspections, reviews, analyses, tests);
- alleviations that can be employed to reduce the severity of failure modes (e.g., array bounds checking coupled with appropriate responses).

Typical DDP applications have involved 10–20 experts drawn from the disciplines of mission science, project planning, software and hardware engineering, quality assurance, testing, risk management, etc.

DDP's particular strength is that it can combine inputs from this wide variety of disciplines. It uses its relatively simple risk-based quantitative model to do so. Certainly this model is incapable of capturing all the nuances of a complex design. However, for early

decision making, it is more important to be able to make key choices: those which if done correctly will lead to significantly superior designs. By seeking to be all encompassing of the relevant areas of expertise, DDP is able to avoid pitfalls of too narrow a focus on just the areas that are understood in depth. The simplicity of the quantitative risk model means that all areas can formulate their concerns to at least a coarse level of fidelity, which is often all that is needed to make key decisions.

3.2 DDP process

There is a straightforward stepwise process to building and using a DDP model. The steps are described below. It has been typical to require at least four sessions of 3–4 h each to gather the DDP information for a non-trivial technology. A facilitator is needed to direct these sessions. This must be someone who both understands the DDP process, and has a feel for the broad range of concerns that the study must deal with. The facilitator guides the elicitation and decision-making steps. The DDP tool is run throughout the sessions, its screen projected and visible to all the participants. As information is gathered, it is entered into the tool in real time. Someone conversant with the DDP tool controls the tool, does data entry, switches between the various visual presentations, etc. In some studies, the same individual has acted as both facilitator and tool controller; in others, separate individuals have filled these two roles.

3.2.1 DDP process step 0: overview the problem and the DDP methodology

This introductory step is to make the participants aware of the purpose of the exercise, familiarize them with the system/technology under study, and explain to them the DDP process itself.

3.2.2 DDP process step 1: develop the impact (requirements \times failure modes) information

- Step 1a: develop the Requirements tree. The pertinent requirements are gathered and organized hierarchically into a tree structure. The leaf requirements need to be “weighted” to reflect their relative importance. The leaves of the tree are the individual requirements whose attainment will be summed to yield the “benefit” measure of total requirements attainment.
- Step 1b: develop the Failure Modes tree. The pertinent Failure Modes are all the things that could possibly occur to adversely impact attainment of the requirements gathered in step 1a. Failure Modes are gathered and organized hierarchically into a tree structure, the leaves of which are the individual Failure Modes.

- Step 1c: relate the leaf Requirements to the leaf Failure Modes. Each Requirement is to be related to the Failure Modes that, should they occur, would adversely impact that requirement. The relationship is assigned a quantitative value, the “impact”.

It is usual practice to follow a methodical approach to elicit these impact values. For example, loop through the Requirements one by one, for each Requirement looping through the Failure Modes one by one to determine which could possibly impact that Requirement, and by how much.

3.2.3 DDP process step 2: develop the effect (PACTs \times failure modes) information

- Step 2a: develop the PACTs tree. The pertinent PACTs are all the activities that could possibly be performed to reduce the likelihoods and/or severities of the Failure Modes gathered in step 1b. Again, these are gathered and organized hierarchically into a tree structure, the leaves of which are the individual PACTs.
- Step 2b: relate the PACTs to the Failure Modes. Each PACT is to be related to the Failure Modes that it reduces or induces. The relationship is assigned a quantitative value, the “impact” value for those it reduces, and the “PACT-induced failure” value for those it induces. Like impacts, it is usual to follow a methodical approach when eliciting these values.

3.2.4 DDP process step 3: decision making

Steps 1 and 2 populate the DDP model. In this final step the populated model is used to guide decision making, which can have multiple objectives:

- Judicious selection of PACTs. The primary decision-making objective is usually the selection of PACTs, balancing their costs (what it will cost to perform them) against their benefits (how much they reduce Failure Modes, and thereby lead to increased attainment of Requirements). The end result will be a judicious selection of PACTs, accompanied by a clear understanding of the purposes of those PACTs (namely, the Failure Modes that they reduce).
- Requirements triage and reprioritization. The model can serve to identify problematic requirements, namely those that the model shows to be at risk from Failure Modes, the reduction of which is expensive. The experts assembled can use the DDP model to guide their selection of which requirements to abandon, or whether to reprioritize the requirements to better match the resources at hand. Alternately, they may use the model to justify the need

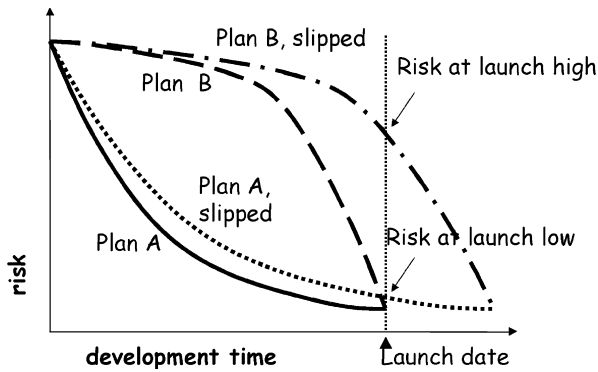


Fig. 2 Risk profiles

for additional funding to improve requirements attainment.

- Comparison of alternatives. Major alternatives can be compared to see what it would cost to attain a certain level of Requirements attainment for each, and to compare the kinds of risks that each alternative is most vulnerable to. When dealing with novel technologies, we have found it useful to categorize Failure Modes into those raised specifically by the novel technology, and those that we regard as “standard” concerns. The former category represents a greater uncertainty, and can hence be a key differentiator between alternatives. Another key differentiator is the way in which the risk profile decreases over the time phases of a planned effort. There is a preference for risk profiles that show early, rather than late, reduction in risk, even if they lead to the same end-point. The reason is that early decision-making models contain considerable uncertainty. A plan that reduces risk early can slip and still have reduced risks to tolerable levels by the originally planned launch date (Plan A in Fig. 2). The same tolerance to slippage is not true of a plan that reduces risk late (Plan B in Fig. 2).
- Gap analysis (a.k.a. bottleneck analysis). The absence of PACTs to address key Failure Modes (ones that significantly impact the Requirements, but for which there are few, if any, PACTs to sufficiently reduce them) becomes apparent from a DDP model. This information points the way to areas in particular need of future study, for example a focused research task to find ways to reduce key Failure Modes.

3.2.5 DDP tool support for process

The DDP tool provides modest support for users to follow the process outlined above. The main “roadmap” screen, shown in Fig. 3, is a simple GUI through which the user can get status overview, process guidance, and process assistance:

- Current status is indicated by showing the numbers of various items (e.g., Requirements) present in the model.

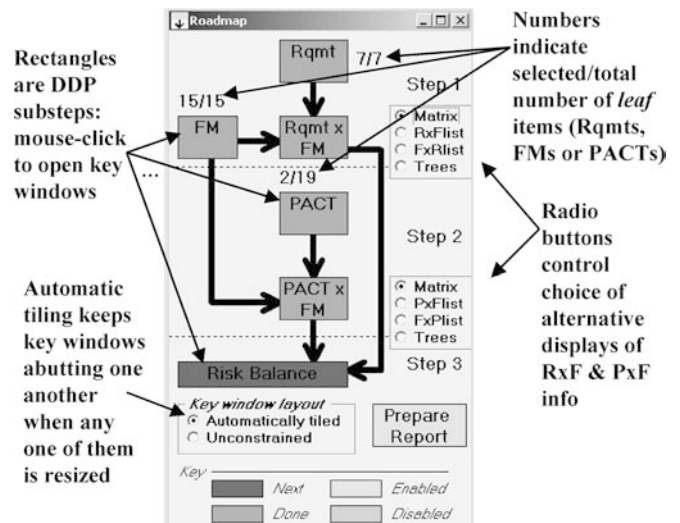


Fig. 3 DDP roadmap

- Guidance is available through color-coded indications of the available and recommended next process steps.
- Assistance is in the form of clicking on a rectangle and having the tool bring up the views relevant to performing that substep (the radio buttons indicate choices among major alternative presentations for a given substep). For example, clicking the “Rqmt” box leads to a window layout of these three inner windows appropriate for entry and scrutiny of requirements:
 - tree view of Requirements, with Requirements’ weights visible;
 - property editor (for examination and editing of attributes of a requirement); and
 - bar chart view showing current attainment status of each of the leaves of the Requirements tree.

The DDP tool enforces process only to the extent of data entry validation (e.g., the tool requires that the a priori likelihood value assigned to a Failure Mode be a number in the range 0 to 1). The process steps outlined above are recommendations, not commandments. In reality, such flexibility is necessary to allow users to deviate from the perfect process, in which everything is dealt with in exactly the expected order.

For example, suppose that during elicitation of Effect values (between PACTs and Failure Modes) users are stepping through the PACTs one by one when partway through they realize that they are missing a Failure Mode. They must add the missing Failure Mode to the Failure Modes tree, revisit the previously considered PACTs to elicit possible Effects of those PACTs on the new Failure Mode, revisit all the Requirements to elicit possible Impacts of the new Failure Mode on those Requirements, and then return

to the Effect values to continue from where they were. These steps form a mini-process that the tool could (but does not presently) explicitly support. Instead, the facilitator must know to follow a methodical approach that is appropriate.

3.2.6 DDP process alternatives

There are ways of approaching the information elicitation process other than the start-from-an-empty-sheet implicit in the standard DDP process – for example, use of a knowledge base of known Failure Modes and PACTs as a starting point. Indeed, we have built some knowledge bases of such knowledge for various spacecraft hardware and software concerns. The process by which users start from these pre-existing knowledge bases of DDP information is quite different, and in need of its own process support. This is an area of active investigation. In the area of software assurance planning, we have linked the DDP tool to NASA's Ask Pete tool. The latter does estimation and planning of software assurance activities. In our combination of these, the Ask Pete tool is used to build a first-cut model, and the DDP tool can then be used to tailor this to the development at hand [1]. Within our group Julia Dunphy is working on an "interview"-like front-end that queries the users for the broad characteristics of their mission (e.g., earth orbit, inner planets or outer planets? orbiter or lander?) the answers to which direct which subsets of the knowledge base to bring into play. Adjustments to the tool to support these alternative processes (e.g., more support for searching, selecting and culling from pre-existing datasets) will likely be needed.

3.2.7 Emergent good practices

There are skills to using DDP that we have developed but not codified in any formal process. We have found that in eliciting information from a group of experts, we can use disagreement to guide the need for refining the information. For example, if there is disagreement about the impact of a Failure Mode on a Requirement, this almost always stems from those experts thinking of different cases (e.g., the impact in the "nominal" scenario, vs. the impact in a high-criticality scenario). Subdividing the Failure Mode and/or the Requirement into multiple subcases, and assigning appropriately different impact values to each, resolves these disagreements.

Similarly, agreement indicates the lack of need to subdivide into greater depth. We do not always recognize the latter in advance, resulting in subcases that we find are being assigned the same values. When this occurs, we simply delete the myriad of subcases, and make do with the parent. This gives us confidence in the results, reminiscent to performing a calculation to $N+1$ significant digits in order to emerge with an answer that

we know is correctly rounded to N significant digits. It is, however, wasteful of time. Given that a DDP session involves the simultaneous presence of multiple experts, we are motivated to make the best use of their time.

Overall we see as key the drive towards a common understanding, and an identification of distinctions when these are relevant to the decisions being made.

3.2.8 DDP results

We have not performed any formal experiments to measure the overall effectiveness of DDP applications. We know the cost: predominantly the time of the experts involved in populating the study with data and making decisions. For example, a typical DDP application, with 15 participants in all four of the 4-h sessions, consumes 240 h (6 work weeks) of time. We require the participation of experts. Such people are always in demand, so this is by no means a trivial investment of time.

Cornford et al. [2] reported outcomes of DDP applications:

To date, we have applied DDP to four component level developments and one software development with extremely successful outcomes. In one case, the clarification of a major customer requirement led to ~\$1.2 M savings in work not required, and a product delivery to the customer two years earlier. In another case, a technology targeted for termination due to a lack of customer interest, and poor hope for success, was rescued and is now proceeding to multiple customer utilization. The software study led to consideration of a commercial software development environment to replace the expensive software design practices used at NASA today. Another technology development was discovered to be a hopeless waste of funding given its progress, status, and team attrition situation.

Lacking any formal study, it is impossible to attribute these beneficial outcomes solely to DDP. Since then, the continued voluntary use of DDP (it is not mandated by JPL's development principles) on further applications, and in recent months on a large-scale project, is at least indicative of its net value.

4 DDP information structures

DDP's fundamental information structures to represent the model are nodes (organized into hierarchical trees) and links. Attributes of nodes and links hold further information on each.

4.1 DDP nodes and links, and their attributes

There are three main types of nodes (Requirements, Failure Modes, and PACTs), organized into hierarchical

trees, and two types of links (Impacts, between Requirements and Failure Modes, and Effects, between PACTs and Failure Modes).

Individual nodes and links have attribute values, some of which are common to all (e.g., the “Description” and “Notes” attributes which are unbounded text entries), some of which are common to nodes (e.g., the “Title” attribute, a text string of bounded length), some common to links (e.g., the “Value” attribute, for an Impact link its Impact value, for an Effect link its Effect value, both of which are floating point numbers in the range 0–1 or a bounded-length text string beginning with certain alphabetic characters, essentially a comment for use when a numerical value has not yet been provided), and some of which are specific to the type of node or link (e.g., the “Weight” attribute of a Requirement).

The nodes are organized into hierarchical trees, such that only the leaves of those trees are the nodes pertinent to the calculations. The non-leaf nodes act like folders in a file structure, as containers for nodes or further containers.

4.1.1 Information structures

The simplicity of the model allows DDP to accommodate a wide range of issues. For example, both product and process requirements can be included in the same model. Users have the flexibility to make as much, or as little, use of the attributes as they desire:

- The minimalist extreme has been simply titles for all three types of nodes and a single kind of marker in the value attribute of Impacts and Effects to indicate that there is *some* impact or *some* effect. In this mode, DDP acts in a purely qualitative manner to implement the “Risk Balancing Profiles” idea of Greenfield [3].
- For using DDP to quantitatively calculate benefits (but not costs), users need to provide titles for all three types of nodes, weights for Requirements, a priori likelihoods for Failure Modes, and numerical Impact and Effect values. Attributes such as the textual description are not necessary, but prove useful as a place to record more explanation (e.g., to help understanding when revisiting the model at a later date, or when other people read a model, akin to comments in code). To date this has been the most commonly adopted style of using DDP. The lack of cost information has meant that users must mentally estimate costs when selecting PACTs, rather than relying on the tool to do so.
- For using DDP to quantitatively calculate both benefits and costs, users need to provide cost information in addition to the benefit information listed above. At a minimum, they must cost the individual PACTs. To encompass the ramifications of the cost of detection as different from the cost of

repair, users must also provide repair costs for Failure Modes, and indicate the type of each PACT (detection, prevention, or alleviation).

Once the model contains both cost and benefit information, automated support to search for (near) optimal solutions becomes feasible. We have had success using a machine-based learning approach [4], and in addition we are pursuing the use of genetic algorithms for this same purpose (discussed in [5]).

One purpose of the hierarchical organization of nodes is to aid navigation; as the number of requirements increases into the tens and hundreds, finding where a requirement is located becomes easier if they are logically grouped. It also supports inspiration: a category may serve as a reminder of a whole class of requirements to consider. Finally, the DDP tool exploits this same hierarchical structure to give users control over the level of detail they see. When they “collapse” a subtree (akin to closing a folder in a file structure), the tool automatically computes the aggregate of the leaves of that subtree and uses that aggregate value to serve as an abstraction of the details beneath. Views within the DDP tool itself are automatically coordinated so that the user sees the same hierarchy status in each.

An example is shown in Fig. 4, where within the Failure Modes tree (left side) “Functional Operation Issues” is a collapsed subtree. Visual cues are the tiny “+” box to the left, and the “closed” folder icon. In the Requirements x Failure Modes matrix (right side) is a column corresponding to this collapsed subtree. Visual cues are the “[+]” in its header cell, and the gray background, indicating that the values therein are computed. For example, the highlighted cell between the “Get to the target” Requirement and the “Functional Operation Issues” Failure Mode has a value of 4.1, computed as the sum of the impact values between the leaf Failure Modes of that entire subtree against the Requirement.

4.1.2 Information compromises

There are many ways in which the DDP model’s simple information structures compromise the representation of the actual situation (system or technology under consideration). We list the more notable of these:

- Uncertainties. Wherever a quantitative value is required, DDP allows for only one value to be provided. In practice, information will be uncertain, and it would perhaps be desirable to record this uncertainty in the model and be able to reason about it. Towards this end, we are contemplating extending the representation of costs to accommodate a trio of values: optimistic (e.g., 10th percentile), median, and pessimistic (e.g., 90th percentile). The similar treatment of impact and effect values is

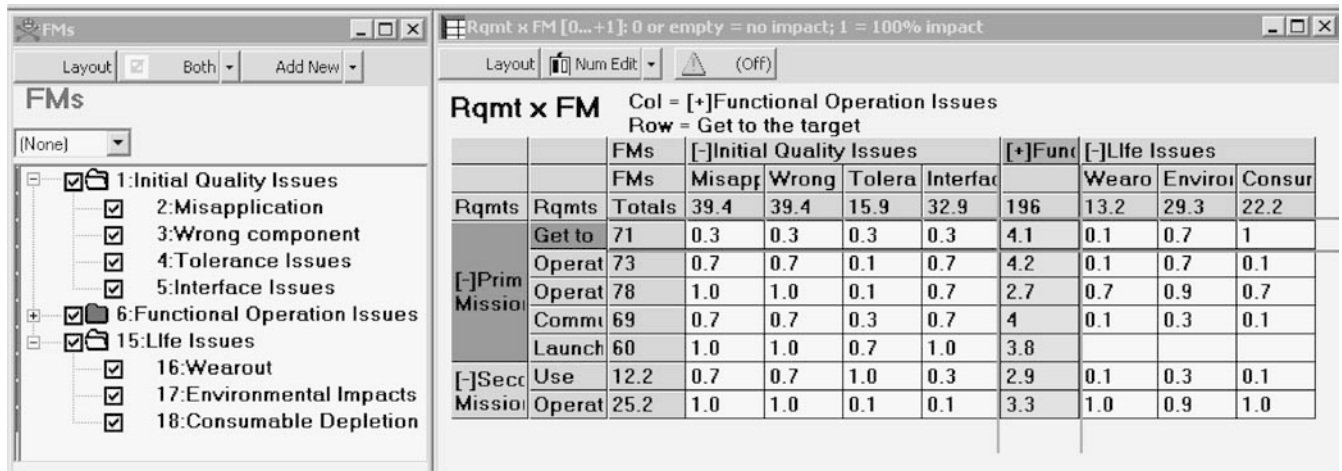


Fig. 4 Hierarchy and aggregate values

also possible, although we are concerned that this will significantly add to the amount of work that is needed to populate the DDP model. The number of cost values to ask for is linear with the number of PACTs and Failure Modes (for repair costs). However, the number of impacts and effect values is in proportion to the *square* of the number of nodes (Requirements, Failure Modes, and PACTs), so clearly would be a much greater additional burden during model population.

- Interactions between elements. The simple combination rules for impacts and for effects are a compromise. In practice there are situations where certain combinations will have costs/effects/impacts that differ significantly from the value our model would compute. In the absence of the capability to represent and reason about these situations, we currently rely on manual workarounds. For example, when we know that the combination of two PACTs M1 and M2 does not match that predicted by our formulae, we manually add a third PACT, M1&M2. We assign to this the combined effectiveness and cost values that we believe hold for the combination of the two. When selecting PACTs, we are careful to select at most one of {M1, M2, M1&M2}. Such workarounds allow us to proceed with DDP applications, at the expense of a small amount of additional effort.
- Logical structure to failure modes, for example, the and/or nodes used to built fault trees as used in probabilistic risk assessment. DDP's cumulative calculation of impact is a crude but useful numerical approximation of "or" nodes in a fault tree. For example, if one of the Failure Modes is "Explodes on Launch", then most mission requirements will be totally lost should this occur. We model this by giving this Failure Mode an impact score of 1 against those requirements. "And" nodes are more problematic, and require the kind of workaround

described in the previous paragraph. We recently added and/or logical structure to DDP's failure modes, but have not yet had a chance to exercise these capabilities in a DDP application.

Generally, our approach to dealing with these and other similar information compromises is to elaborate the DDP model if and when we find recurring need to do so. As we continue to expand the variety of problems that we study with DDP, we expect to have to continue the evolution of the DDP information representation, and the tool that supports its application.

5 Information visualization

DDP's information visualization needs derive from DDP's purpose, to *assist* experts in their decision-making concerning novel and complex systems. The experts provide the information that goes into a DDP model, and the experts make decisions based on that model together with their further knowledge that has not necessarily been incorporated into the model. Feedback through appropriate visualization is key. It is also challenging. For the advanced technologies and systems that we deal with, decision making in even the early phases commonly involves more information than can be conveniently portrayed on the screen at once. (If there were so little information that it could be portrayed on a single screen, there would hardly be any need for a special process or tool.)

In response to these needs and challenges, DDP gives users a choice of visualizations, and modest layout control over the individual visualizations. The visualizations themselves are composed of relatively standard elements (matrices, lists, tree structures, bar charts). Where possible these mimic the look and feel of the familiar Microsoft Windows controls. Simple GUI design principles are followed, such as uniform cues to the users throughout the tool. For example, the use of "+" and "-" boxes or their ASCII approximations "[+]" and "[-]" denote collapsed/expanded subtrees. Color

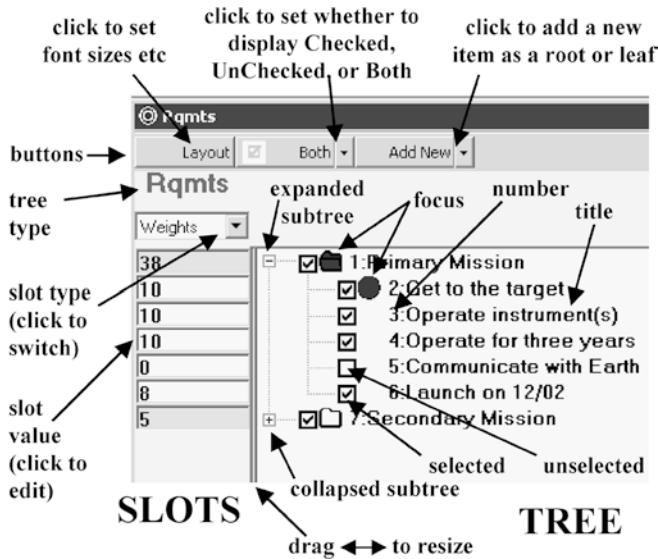


Fig. 5 DDP tree view

conventions such as uniform use of blue for Requirements, red for Failure Modes, and green for PACTs apply across most of the tool (the user can change this default choice of colors).

5.1 Visualization for node hierarchies

The primary visualization for node hierarchies is the Tree View, shown in Fig. 5 (annotated with “help” information). Its visual cues include:

- “+” and “-” boxes, together with closed/open folder icons to indicate collapsed or expanded sub-trees;

- blue color (reproduced here as light gray) to remind the user this is a tree of Requirements;
- white/gray backgrounds of the slot values to denote user-editable/computed values. For example, the Weight of the Primary Mission Requirement, 38, is computed as the sum of the weights of the leaf requirements within its subtree.
- colored-in items to indicate the current “focus” on a particular requirement (number 2, “Get to the target”, alongside which is a blue circle – reproduced here as a light gray colored circle) and its heritage (child of number 1, “Primary Mission”, alongside which is a blue filled-in folder – reproduced here as a light gray filled-in folder).

Users control over the layout includes:

- selection of the kind of Requirements attribute shown in the slots (in Fig. 5 it is set to show the Weights attribute values);
- whether to view just the checked (a.k.a., “selected” Requirements), the unchecked ones, or both (in Fig. 5 it is set to show both);
- selection of font size, etc. This seemingly trivial feature is very useful to allow the user to tune the DDP display to the situation. For example, when using DDP in a group setting, the image is projected on a screen visible to all, and so the font size has to be set to make the text legible to all. Alternately, when using DDP on a single workstation, a user may wish to set the font size to the smallest that he/she can read, so as to squeeze as much information as possible onto the screen at once.

In the DDP process a significant amount of time is spent working with this tree view. It is central to

Fig. 6 DDP matrix view

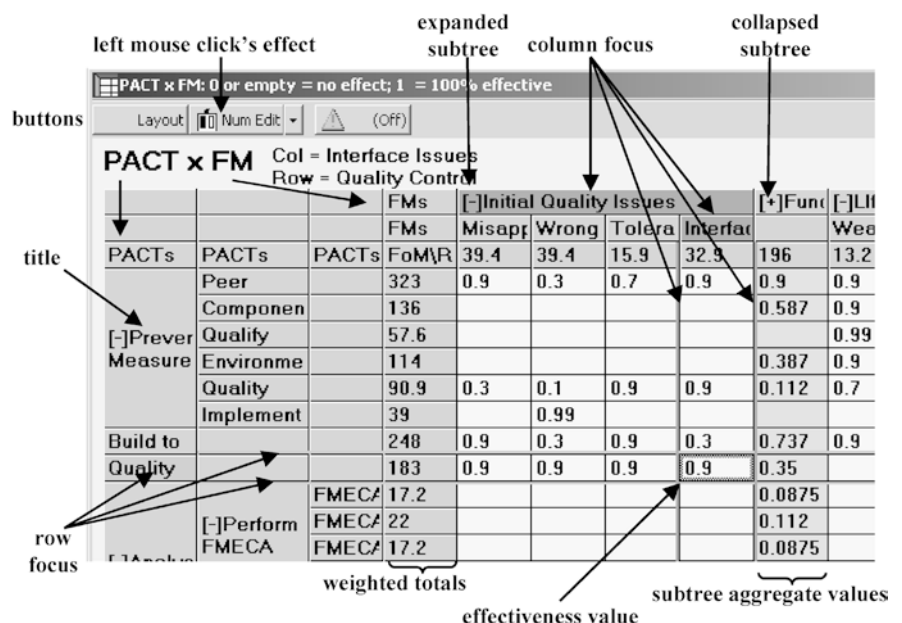
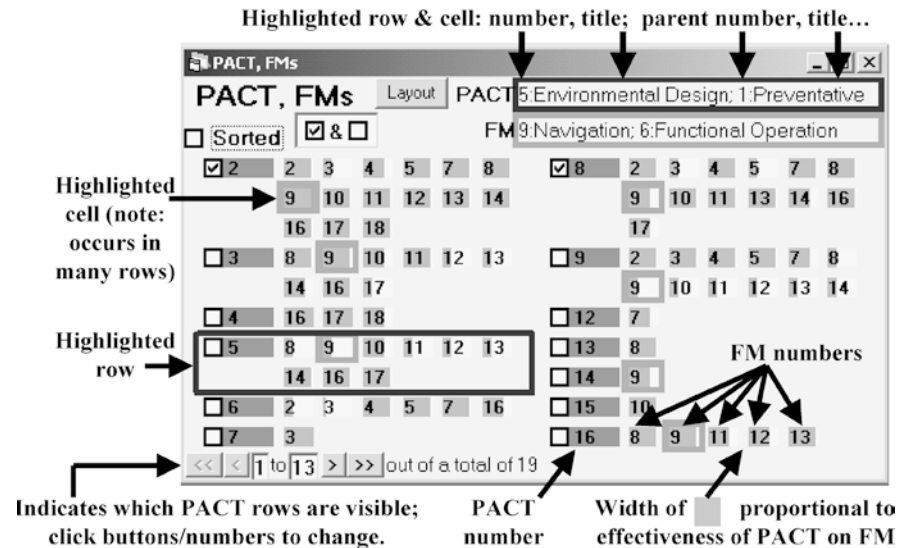


Fig. 7 DDP stem and leaf view



developing DDP's three trees, of Requirements, Failure Modes, and PACTs.

3.2 Visualizations for links

DP information includes Impact links between Requirements and Failure Modes, and Effect links between PACTs and Failure Modes. DDP offers two main visualizations for these links: a "matrix"-like presentation (Fig. 6) and a "stem and leaf"-like presentation (Fig. 7). Motivation for the use of each derives from DDP's leaning towards simplicity and volume. DDP's links are between only pairs of nodes (there are no three-way links, or links connected to other links, for example).

While each DDP link can have multiple attributes (e.g., Description, Reference) associated with it, of these only the Value (Impact or Effect) is essential to display for multiple links at once. Typical DDP models contain possibly hundreds of nodes of each of the three main DDP types (Requirements, etc.), so the number of links of a given type can be in the thousands.

The matrix view is good for showing the correspondence between the link values (shown in the inner cells of the matrix) and the nodes they link (shown in the hierarchical header row and column cells of the matrix). However, in practice we find that DDP models have relatively sparse matrices (ranging from 5% to 30% filled), for which matrices are wasteful of space (lots of blank cells).

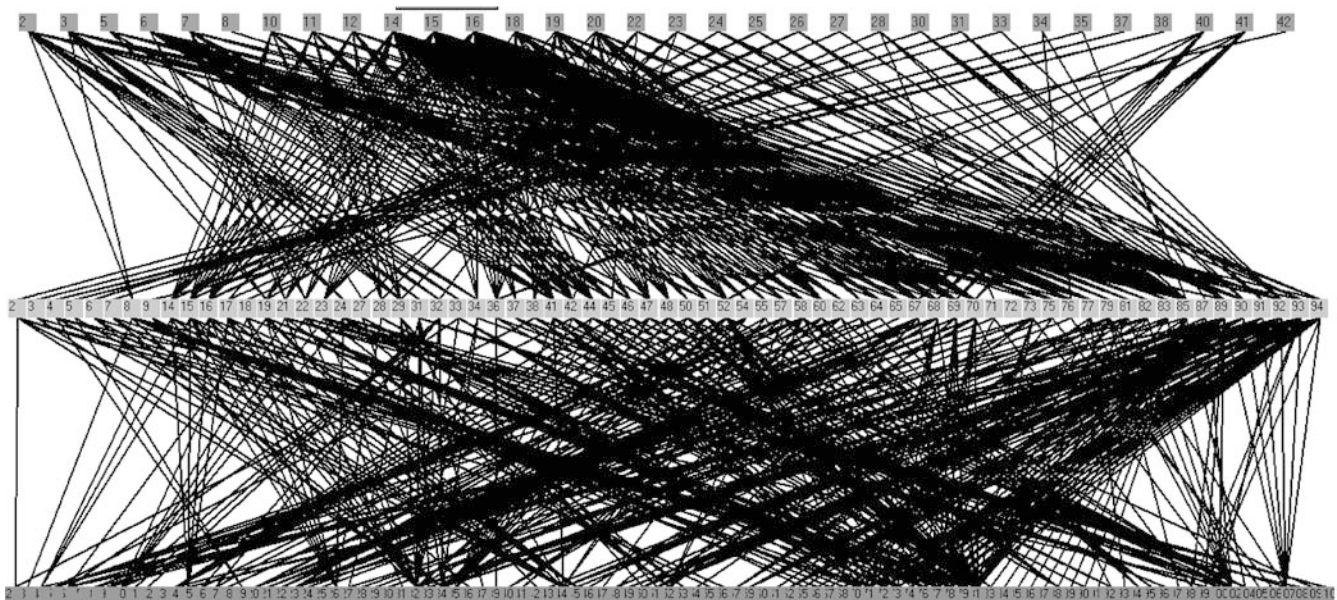


Fig. 8 Topology of a DDP model

Fig. 9 DDP bar chart of failure modes

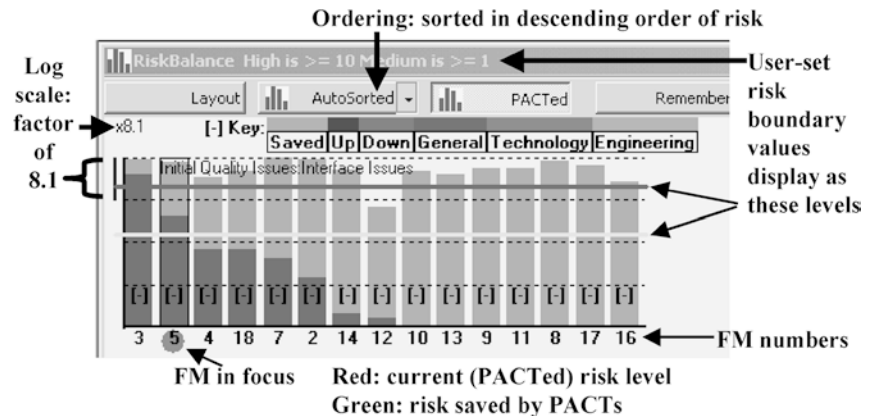
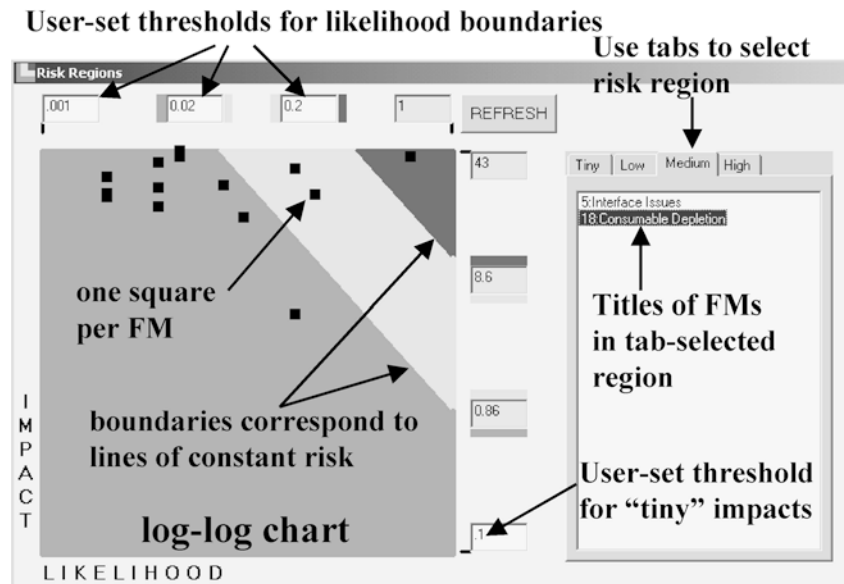


Fig. 10 DDP 2-dimensional plot of failure modes as risks



The general form of a “stem and leaf” style of information presentation is described in Tufte [6], who attributes this form of plot to Tukey [7]. At JPL, Denise Howard (JPL) and Christopher Hartsough introduced its use in their prototype of the “Risk Balancing Profiles” (RBP) concept [3]. Roughly speaking, RBP is a subset of DDP: it has the equivalent of just DDP’s Failure Modes and PACTs. Furthermore, RBP is purely qualitative. Its Failure Modes’ severities are user-assessed qualitative values (rather than being derived from their aggregate impact on requirements), and its links between PACTs and Failure Modes indicate only the existence of reducing effects, not their magnitudes. This was seen as a natural precursor to DDP, and we arranged to permit RBP’s information to be transferable into DDP [8].

We adopted RBP’s stem and leaf style of presentation for DDP, elaborating it to indicate DDP’s qualitative information. Effects links are shown portrayed in this manner in Fig. 7. The numbered wider rectangles form the “stem” of the display, each corresponding to

a PACT. Alongside each are numbered smaller rectangles, corresponding to the Failure Modes that PACT effects. Quantitative information is conveyed by the width of the smaller rectangle, proportional to its link’s Effect value.

This view is good as a concise display of a large number of the extant links, but compared to the matrix view is less successful at displaying the hierarchies of the nodes being linked. We do *not* make use of the more free-form graph views common to many other requirements engineering methods and tools (e.g., the hypertext graphs of gIBIS [9]; the class diagrams of UML; the influence diagrams of i* [10]; goal graphs [11]). These seem better suited to models that are smaller but more topologically complex than DDP’s. Figure 8 shows a DDP model drawn using explicit links; while the layout is naïve (there has been no attempt to reorder nodes to disentangle the links), it does serve to convey the nature of DDP models as being simple in structure but verbose in content.

5.3 Visualizations for results of DDP calculations

Visualizations of the results of DDP's calculations are available to support the decision-making step of the DDP process (Section 3.2.4). These include a simple display of overall cost and benefit figures, bar chart displays to indicate status of Requirements, Failure Modes and PACTs, and two-dimensional plot of the risk (severity x likelihood) status of Failure Modes.

Figure 9 shows the bar chart (histogram) display for Failure Modes. Each bar corresponds to a Failure Mode. The height of the red portion (reproduced here as dark gray) of a Failure Mode's bar indicates its sum total impact on Requirements, taking into account the effect of currently selected PACTs. The top of the green portion (reproduced here as light gray) indicates where its sum total impact level used to be, when no PACTs were selected. Thus the green portion corresponds to the risk savings of the current selection of PACTs, and the red portion the currently extant risk of that Failure Mode. Sorting in descending order of the extant risk (i.e., in decreasing height of red portions) is a Pareto-like activity, allowing users to focus their attention on the most significant remaining risks. Also, it prevents accidentally overlooking significant risks in a large population of risks.

Figure 10 shows a two-dimensional plot of Failure Modes as risks. The axes of the plot are likelihood and severity (each a *log* scale); Failure Modes (represented by small black squares) are located accordingly. This gives a more compact view of the Failure Modes, and makes visible their breakdown into likelihood and severity. (Note, however, that it compromises the DDP tool color convention where red is to represent Failure Modes, and green PACTs; instead, this chart follows the common "traffic light" pattern of using red (the upper right triangular region, reproduced here as dark gray) / yellow (the middle band, reproduced here as light gray) / green (the large lower-left region, reproduced here as an intermediate gray) to denote risk categories of decreasing magnitude. Of course, most Failure Modes make their way from the red area to the green area thanks to PACTs, so perhaps this is not so divergent after all.)

5.4 Visualization challenges

The DDP visualizations significantly aid users in their decision-making tasks, by presenting the status of a DDP model (overall costs and benefits, and the status of individual elements such as Failure Modes).

However, much of the decision-making rests on the comparative costs and benefits of PACT selections. None of the visualizations of the current DDP tool attempt to display this selection space. To do so is a daunting task: consider that for N PACTs, there are 2^N such selections, and a typical DDP model has dozens, possibly hundreds of PACTs.

There is clearly a need for further work to connect visualization with the automated search for near-optimal PACT selections. Some preliminary steps in this direction are the plots in Feather and Menzies [4].

6 Related work and conclusions

The DDP model has some similarity with a number of other models of systems. We briefly discuss several of these below.

6.1 Qualitative decision support methods

Early decision making is often assisted by qualitative decision support methods. Quality Function Deployment (QFD) is prominent among these, and has been used in a wide variety of settings [12]. DDP's effect and impact matrices are reminiscent of the Relationship Matrix used in many forms in QFD.

DDP is distinguished by its foundation upon a *quantitative* risk model, which gives meaning to DDP's cost and benefit calculations.

6.2 Estimation models

COCOMO and COQUALMO are models to predict factors such as cost and quality based on inputs that characterize the development at hand [13]; the stochastic model in Stutzke and Smidts [14] is similar.

Generally, estimation models such as these are "closed", in the sense that they are not intended to be extended with new factors (although they do encourage the tuning of these models to a given organization). In contrast, the DDP model is "open", relying on expert users to input and link the factors that are relevant to the development at hand during DDP sessions. We have explored a mix of these approaches, by linking the DDP tool to NASA's Ask Pete tool. The latter does estimation and planning of software assurance activities. In our combination of these, the Ask Pete tool is used to build a first-cut model, and the DDP tool can then be used to tailor this to the development at hand [1].

6.3 Goal models

The software engineering research community has shown increasing interest in models of "goals" (roughly speaking, precursors to requirements). See the mini-tutorial [11] for an overview of this area. We discuss two of these kinds of models:

The KAOS framework for goals, requirements, etc. [15] is used to build a logical structure of how

system-wide requirements decompose to, ultimately, requirements on the individual components in a system. Models built in this framework seem well suited to exploring the functional behavior and, to some extent, non-functional aspects. DDP models are weaker in that they lack the logical structure of KAOS models, but conversely have emphasized more the quantitative aspects that predominate in imperfect solutions.

The i* framework [10, 16] combines logical structures with *qualitative* models. Their combination rules for these qualitative models support well tradeoff analysis between major design alternatives. DDP models seem more appropriate when there are a large number of small alternatives.

6.4 Bayesian nets/influence diagrams

Influence diagrams (a form of Bayesian nets) offer a general framework in which factors can be combined to assess designs and study alternatives. For example, they are used in Burgess et al. [17] to compute the utility of requirements that are candidates for inclusion in the next release of a piece of software. In principle it would seem that a DDP model could be represented as an influence diagram, with a relatively “flat” topology. However, as was seen in Fig. 8, typical DDP applications give rise to rather voluminous such models. DDP is perhaps more appropriate to decision making when a multitude of factors must be considered.

6.5 Requirements prioritization

Requirements prioritization has also emerged as a topic of interest within both academia and industry. Karlsson and Ryan developed a “cost-value” approach to prioritizing requirements [18]. At the heart of their approach is a cost-value diagram, which plots each requirement’s relative value and implementation cost, facilitating the selection of an appropriate subset of requirements.

The WinWin project [19] supports multiple stakeholders to identify conflicts between their respective evaluations of requirements, and to locate feasible solutions that are mutually satisfactory combinations of requirements. This approach is supported by a custom tool, the benefits of which are reported in Boehm [20].

These examples typify approaches in which users are asked to directly estimate the costs and benefits of individual requirements. Significant interactions among requirements, for example, if two requirements can be achieved by sharing the same solutions to sub-problems, complicate this task. DDP approaches this problem by explicitly relating requirements to failure modes, and in turn failure modes to PACTs.

6.6 Probabilistic risk assessment

The term “Probabilistic Risk Assessment” (PRA) encompasses a mature and widely used set of techniques for conducting quantitative risk assessments. For example, the fundamentals of fault trees are described in Vesely et al. [21]. Originating in the nuclear power industry, PRA techniques are now regularly applied in nuclear, process, chemical, petroleum, aerospace, and other industries. Generally these approaches presuppose the existence of a detailed design. One of the challenges for PRA techniques is application early in the life cycle, most especially in the formative stages of development, when the system design is incomplete, immature and/or still in flux. DDP is intended for this kind of situation. Furthermore, DDP emphasizes not just risk assessment, but also planning for what can be done to reduce risk.

6.7 Conclusions

We have outlined the DDP model, designed to fill the early-life cycle niche in failure mode-based estimation and planning. DDP thus complements a number of other modeling techniques.

DDP involves just three key concepts: “*Requirements*” (what it is that the system or technology is to achieve), “*Failure Modes*” (what could occur to impede the attainment of the Requirements), and “*PACTs*” (what could be done to reduce the likelihood and/or impact of Failure). Requirements are qualitatively related to Failure Modes, and Failure Modes are in turn qualitatively related to PACTs.

Custom tool support facilitates the use of this model when relatively large numbers of items are involved, and a simple process guides the overall application of DDP.

DDP has been successfully applied in early life cycle decision making. It appears well suited to applications where a multitude of factors must be considered simultaneously. We fully expect use of DDP to continue. We also anticipate that the DDP model will continue to evolve in response to the needs of new applications.

6.8 DDP further reading

We have published several accounts of DDP in conference and workshop papers.

For an overview of DDP and its applications, see Cornford et al. [2]. More recent directions are described in Cornford et al. [5]. Application to software development is discussed in Cornford [22] and to software assurance in Feather et al. [23].

Crafting the “look and feel” of DDP to support users’ tasks has been a concern all along. Key decisions are reported in Feather et al. [24] and issues related to scalability are reported in Feather et al. [25].

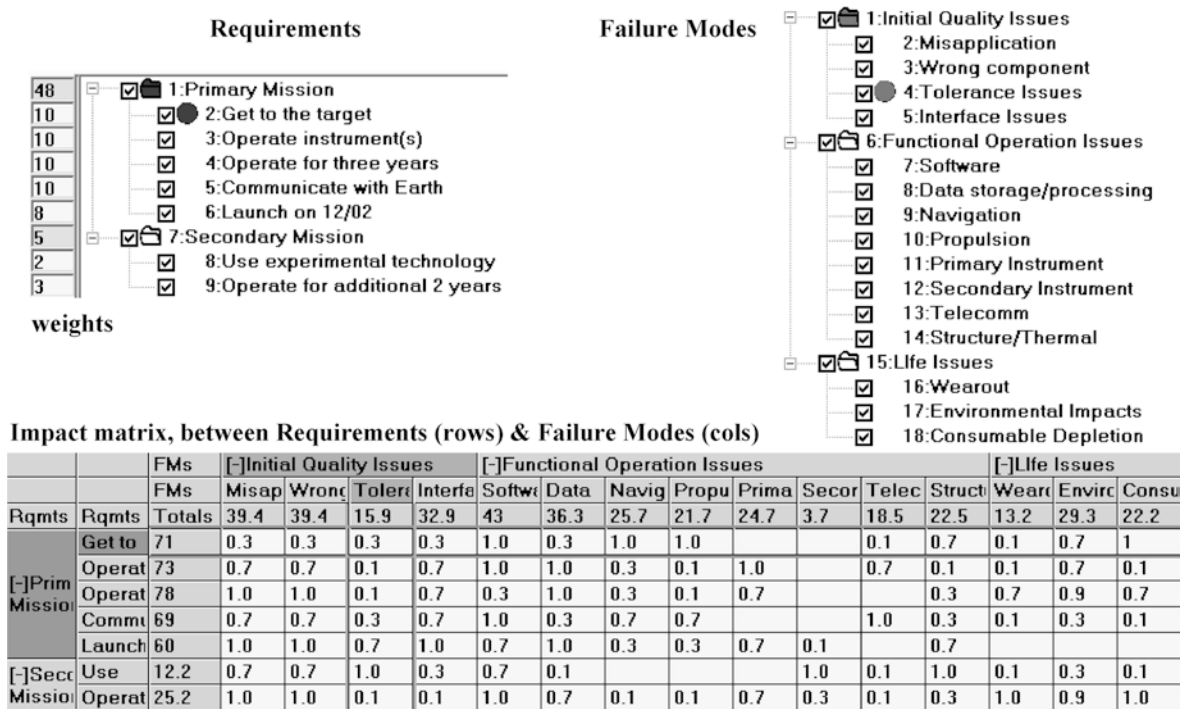


Fig. 11 Requirements, failure modes and impact matrix, between requirements (rows) and failure modes (cols)

The transition from a qualitative model to a quantitative model is explored in Feather et al. [8].

Experiments using Menzies' machine-learning based techniques for automated optimization with DDP models are reported in Feather and Menzies [4].

Up-to-date information on DDP, including the ability to request a copy of the program, can be found at <http://ddptool.jpl.nasa.gov>.

Acknowledgements The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology. Contributions from, and discussions with, Phil Daggett (JPL), Julia Dunphy (JPL), Patrick Hutchinson (Wofford College, Spartanburg, SC), Ken Hicks (JPL), Christopher Hartsough (JPL), Denise Howard (JPL), Peter Hoh In (Texas A&M), John Kelly (JPL), Tim Kurtz (NASA Glenn), James Kiper (Miami University, OH), Tim Larson (JPL), Tim Menzies (University of British Columbia), Kelly Moran (JPL) and Burton Sigal (JPL) have been most useful in helping us formulate our ideas and bring them to fruition.

Appendix: DDP concepts and example

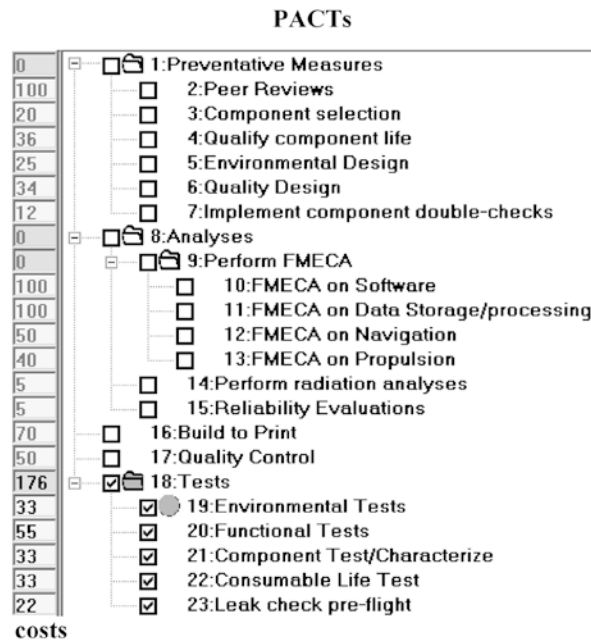
Requirements are whatever the system under scrutiny is to achieve, and operational constraints on the system's construction and operation. Each requirement has a *title*, a *position* in the Requirements tree, a *weight*

(representing its relative importance), *on/off* status, and optional further information such as *description*, *notes*, and *reference*. Only those Requirements whose status is on are taken into account in the qualitative calculations.

Failure Modes are the things that, should they occur, will lead to lack of attainment of Requirements. Each Failure Mode has a *title*, a *position* in the Failure Modes tree, an *a priori likelihood* (the chance of the Failure Mode occurring, if nothing is done to inhibit it), a *repair cost* per phase (what it would cost to remove an instance of that Failure Mode at that phase in the project), *on/off* status, and optional further information such as *description*, *notes*, and *reference*. Only Failure Modes whose status is on are taken into account in the qualitative calculations.

PACTs are the activities that could be done to reduce the likelihood of Failure Modes and/or reduce their impact on Requirements. Each PACT has a *title*, a *position* in the PACTs tree, a *cost*, the *phase* in which it applies, *on/off* status, and optional further information such as *description*, *notes*, and *reference*. PACTs are classified into *preventions* (which reduce the likelihood of Failure Modes), *detections* (which discover instances of failure modes so that those detected failure modes can be corrected prior to release/use), and *alleviations* (which reduce the severity of Failure Modes). Only those PACTs whose status is on are taken into account in the qualitative calculations, with the exception of calculations specifically to reveal what would be the net effect (in terms of risk reduction) were an "off" PACT to be turned on.

Impacts are the qualitative relationships between Requirements and Failure Modes. Each impact has the Requirement and Failure Mode it links, a *value* representing the proportion of loss of attainment of the



Effect matrix, between PACTs (rows) & Failure Modes (cols)

			FMs	[-]Initial Quality Issues				[-]Functional Operation Issues								[-]Life Issues		
			FMs	Misap	Wronc	Toler	Interfe	Softw	Data	Navig	Propu	Prima	Secor	Telec	Struct	Wear	Envirc	Consu
PACTs	PACTs	PACTs	FoM\R	0.999	1	0.999	0.998	0.996	1	0.999	1	1	1	1	0.999	1	1	0.999
[-]Prev Measu	Peer		137	0.6	0.6	0.1	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
	Compo		87.8						0.6	0.6	0.4	0.4	0.2	0.3	0.4	0.6	0.3	
	Quality		49.3													0.95	0.95	0.4
	Enviroi		86.9						0.3	0.3	0.7	0.3	0.3	0.3	0.6	0.6	0.6	
	Quality		63.6	0.3	0.1	0.6	0.4	0.4								0.6		
[-]Anal	Implem		39		0.99													
	FMECA		17.2					0.4										
	[-]Perfo		FMECA	25.4					0.7									
	FMECA		FMECA	12.8						0.5								
	FMECA		FMECA	15.2							0.7							
[-]Test	Perforr		65.9						0.4	0.4		0.7	0.7	0.6		0.1	0.3	
	Reliab		124	0.4				0.3	0.5	0.4	0.7	0.6	0.6	0.6		0.7	0.4	0.1
Build			173	0.6	0.3	0.6	0.3	0.3	0.7	0.4	0.7	0.2		0.7	0.6	0.4	0.6	
Quality			133	0.6	0.6	0.4	0.5	0.7	0.2	0.3	0.2	0.2	0.2	0.2	0.2			
[-]Test	Enviroi		165			0.7	0.7		0.3	0.3	0.9	0.9	0.9	0.9	0.9	0.3	0.9	
	Funcio		257	0.95	0.9	0.7	0.95	0.9	0.95	0.8	0.3	0.7	0.8	0.7	0.1	0.1	0.1	0.1
	Compo		160			0.9			0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.95	0.9	
	Consur		25													0.3		0.95
	Leak		21.1															0.95

Fig. 12 PACTs and effect matrix, between PACTs (rows) and failure modes (cols)

Requirement should the Failure Mode occur, and optional further information such as *description*, *notes*, and *reference*. The value may be non-numeric, in which case although it shows up on displays it is ignored in the quantitative calculations; the usual use for this is as a placeholder and reminder for further scrutiny (e.g., a value “to be determined”).

Effects are the qualitative relationships between PACTs and Failure Modes. Each impact has the PACT and Failure Mode it links, a *value* representing the proportion of reduction of the Failure Mode should that PACT be applied, and optional further information such as *description*, *notes*, and *reference*. If the value is negative, it denotes an effect of *increasing*, rather than

decreasing, a Failure Mode’s likelihood. As for impacts, the value may be non-numeric, and if so is ignored in quantitative calculations.

We use a hypothetical example for illustration. This avoids any proprietary issues that would arise from reporting one of the actual DDP applications, and permits the use of somewhat smaller amounts of data than would arise in practice. Nevertheless, this example will serve to illustrate the elements of DDP referenced throughout the paper. The figures are annotated fragments of screenshots taken from the DDP tool running on this example.

Figure 11 shows the Requirements tree, the Failure Modes tree, and the matrix of Impact values between these trees’ elements. The blue coloring (reproduced here as the darker gray border of the highlighted row and background of its header cells) highlights one of

the Requirements, “Get to the target”, and the red coloring (reproduced here as the darker gray border of the highlighted column and background of its header cells) highlights one of the Failure Modes, “Tolerance Issues”.

The matrix header rows and columns show the titles of the items, and some totals computed by DDP.

The third row down contains values (39.4, 39.4, 15.9, etc.) that are the computed totals of loss of Requirements attainment that each Failure Mode causes. For a given Failure Mode F , this value is computed as:

$F.APrioriLikelihood * \sum (R \in AllRequirements): R.Weight * Impact(F, R)$ where $Impact(F, R)$ is the impact value of Failure Mode F on Requirement R (zero if there is no numerical impact asserted between them). For the highlighted Failure Mode, the calculation is:

$$1 * ((10 * 0.3) + (10 * 0.1) + (10 * 0.1) + (10 * 0.3) + (8 * 0.7) + (2 * 1.0) + (3 * 0.1)) = 3 + 1 + 1 + 3 + 5.6 + 2 + 0.3 = 15.9$$

The third column across contains values (71, 73, 78, etc.) that are the computed totals of loss of each Requirements Attainment caused by Failure Modes. For a given Requirement R , this value is computed as:

$R.Weight * \sum (F \in AllFailureModes): Impact(F, R) * F.APrioriLikelihood$

In the example data, all FM's a priori likelihoods are 1, so for the highlighted Requirement, this calculation is: $10 * ((0.3 * 1) + (0.3 * 1) + (0.3 * 1) + (0.3 * 1) + (1.0 * 1) + \dots) = 10 * (0.3 + 0.3 + 0.3 + 0.3 + 1.0 + 0.3 + 1.0 + 1.0 + 0 + 0 + 0.1 + 0.7 + 0.1 + 0.7 + 1.0) = 10 * 7.1 = 71$

These are the totals for risk in the extreme case that nothing is done to prevent the Failure Modes from occurring. The first requirement has an assigned weight of 10, while its loss of attainment is calculated as 71, indicating that it is more than totally at risk.

This example's tree of PACTs is shown in Fig. 12 (top). PACTs' costs are listed in the column to the left of the tree. PACTs' effects on reducing Failure Modes are shown in the matrix in Fig. 12 (bottom). The green coloring (reproduced here as the darker gray border of the highlighted row and background of its header cells) highlights one of the PACTs “Environmental Tests”, and the red coloring (reproduced here as the darker gray border of the highlighted column and background of its header cells) highlights one of the Failure Modes, “Tolerance Issues”.

For the sake of illustration, we have checked those and only those PACTs in the “Tests” folder. DDP calculates their combined cost (176) and combined effect on reducing the likelihood of the Failure Modes. For example, the highlighted Failure Mode is effected by three of those PACTs: 0.7 by Environmental Tests, 0.7 by Functional Tests, and 0.9 by Component Test/Characterize. These are all detection-style PACTs, meaning that their effect is to reduce Failure Modes' likelihoods.

For a Failure Mode F , its “PACTed” likelihood, i.e., taking into account effects of PACTs, is computed as:

$F.APrioriLikelihood * (\prod (P \in PACTs \text{ when } P.Status = On): (1 - Effect(P, F)))$

(If there are PACTs that *induce* Failure Modes, then a slightly more complicated formula must be used.)

For the highlighted Failure Mode, and the five checked PACTs in the Tests folder, the calculation is:

$$1 * ((1 - 0.7) * (1 - 0.7) * (1 - 0.9) * (1 - 0) * (1 - 0)) = 1 * (0.3 * 0.3 * 0.1 * 1 * 1) = 1 * 0.009 = 0.009$$

This was the Failure Mode whose (unreduced) total contribution to loss of Requirements attainment we calculated earlier to be 15.9. The corresponding “PACTed” calculation that takes into account the beneficial effects of the selected PACTs, substituting the Failure Mode's “PACTed” likelihood for its a priori likelihood, thus:

$F.PACTedLikelihood * \sum (R \in AllRequirements): R.Weight * Impact(F, R)$

For the highlighted Failure Mode this is $0.009 * 15.9 = 0.1431$, i.e., considerably reduced.

Because these PACTs were all detections, the reduction in likelihood of this Failure Mode is accomplished by repairing (prior to launch of course!) the problems those PACTs detect. DDP takes this into account in computing the sum total costs. Repair costs for a Failure Mode F detected and repaired in phase PH are calculated as: $(F.PACTedLikelihood \text{ prior to PH} - F.PACTedLikelihood \text{ after PH}) * F.RepairCost (PH)$.

In the example data, Tolerance Issues' RepairCost in the test phase is 250, so DDP computes its cost of repair due to these PACTs as: $(1 - 0.009) * 250 = 247.75$. PACTs from the Preventative Measures folder are prevention PACTs, so would avoid incurring repair costs such as these, and leave far fewer such problems for detection and repair.

These fragmentary examples illustrate DDP's core calculations. Their results are displayed to users via bar charts (e.g., Fig. 9) and the risk region plot (Fig. 10) shown earlier, the overall aim being to aid users in their decision making.

References

1. Kurtz T, Feather MS (2000) Putting it all together: software planning, estimating and assessment for a successful project. In: Proceedings of 4th international software and internet quality week conference, Brussels, Belgium
2. Cornford SL, Feather MS, Hicks KA (2001) DDP: a tool for life-cycle failure mode management. In: IEEE Aerospace Conference, Big Sky, MT, 2001, pp 441–451
3. Greenfield MA (n.d.) Risk balancing profile tool. <http://www.hq.nasa.gov/office/codeq/risk/rbp.pdf>
4. Feather MS, Menzies T (2002) Converging on the optimal attainment of requirements. In: Proceedings IEEE joint international requirements engineering conference, Essen, Germany, 2002, pp 263–270
5. Cornford SL, Dunphy J, Feather MS (2002) Optimizing the design of end-to-end spacecraft systems using failure mode as a currency. In: IEEE aerospace conference, Big Sky, MT
6. Tufte ER (1983) The visual display of quantitative information. Graphics Press, Cheshire, CT

7. Tukey J (1972) Some graphic and semigraphic displays. In: Statistical papers in honor of George W. Snedecor. Iowa State University Press, Ames, IA
8. Feather MS, Cornford SL, Larson T (2000) Combining the best attributes of qualitative and quantitative risk management tool support. In: Proceedings, 15th IEEE international conference on automated software engineering, Grenoble, France, 11–15 September 2000. IEEE Computer Society, pp 309–312
9. Conklin J et al (2001) Facilitated hypertext for collective sensemaking: 15 years on from gIBIS. Hypertext '01 conference, Aarhus, Denmark, 14–18 August 2001
10. Chung L, Nixon BA, Yu E, Mylopoulos J (2000) Non-functional requirements in software engineering. Kluwer, Dordrecht
11. van Lamsweerde A (2001) Goal-oriented requirements engineering: a guided tour. In: Proceedings 5th IEEE international symposium on requirements engineering, Toronto, Canada, August 2001, pp 249–263
12. Akao Y (1990) Quality function deployment. Productivity Press, Cambridge, MA
13. Boehm B et al (2000) Software cost estimation with COCOMO II. Prentice-Hall, Englewood Cliffs, NJ
14. Stutzke MA, Smidts CS (2001) A stochastic model of fault introduction and removal during software development. IEEE Trans Reliability 50(2):184–193
15. Bertrand P, Darimont R, Delor E, Massonet P, van Lamsweerde A (1998) GRAIL/KAOS: an environment for goal driven requirements engineering. In: 20th international conference on software engineering, Kyoto, Japan
16. Mylopoulos J, Chung L, Liao S, Wang H, Yu E (2001) Exploring alternatives during requirements analysis. IEEE Software 18(1):92–96
17. Burgess CJ, Dattani I, Hughes G, May JHR, Rees K (2001) Using influence diagrams to aid the management of software change. Requirements Eng 6(3):173–182
18. Karlsson J, Ryan K (1997) A cost-value approach for prioritizing requirements. IEEE Software September/October:67–74
19. Boehm B, Bose P, Horowitz E, Lee M (1994) Software requirements as negotiated win conditions. In: Proceedings 1st international conference on requirements engineering, Colorado Springs, CO, pp 74–83
20. In H, Boehm B, Rodgers T, Deutsch M (2001) Applying WinWin to quality requirements: a case study. In: Proceedings 23rd international conference on software engineering, Toronto, Ontario, Canada, pp 555–564
21. Vesely WE, Goldberg FF, Roberts NH, Haasl DF (1981) Fault tree handbook. US Nuclear Regulatory Commission, NUREG-0492
22. Cornford SL (2000) Design and development assessment. In: Proceedings, 10th IEEE international workshop on software specification and design, San Diego, CA, 5–7 November 2000. IEEE Computer Society, pp 105–114
23. Feather MS, Sigal B, Cornford SL, Hutchinson P (2001) Incorporating cost-benefit analyses into software assurance planning. In: Proceedings, 26th IEEE/NASA software engineering workshop, Greenbelt, MD, 27–29 November 2001
24. Feather MS, Cornford SL, Gibbel M (2000) Scalable mechanisms for requirements interaction management. In: IEEE international conference on requirements engineering, Schaumburg, IL
25. Feather MS, Cornford SL, Dunphy J, Hicks K (2002) A quantitative risk model for early lifecycle decision making. In: Integrated design and process technology, Pasadena, CA, June 2002